Hoare Logic

Andreas Podelski

May 16, 2017

Hoare logic

- introduced by Hoare in 1969
  builds on first-order logic

Hoare logic

- introduced by Hoare in 1969
  builds on first-order logic
- correctness specification = pre- and postcondition pair

Hoare logic

- introduced by Hoare in 1969
  builds on first-order logic
- correctness specification = pre- and postcondition pair
- standard presentation of Hoare logic:
  proof uses invariant for every loop in program

Hoare logic

- introduced by Hoare in 1969
  builds on first-order logic
- correctness specification = pre- and postcondition pair
- standard presentation of Hoare logic:
  proof uses invariant for every loop in program
- here:
  invariants are given as part of correctness specification

Hoare logic

- introduced by Hoare in 1969
  builds on first-order logic
- correctness specification = pre- and postcondition pair
- standard presentation of Hoare logic:
  proof uses invariant for every loop in program
- here:
  invariants are given as part of correctness specification
- correctness proof possible only if invariants are adequate for pre- and postcondition pair

Programs

- (program) expression

$$e \ ::= \ x \mid f(e_1, \ldots, e_n)$$

where f maps into domain of values

Programs

- (program) expression

$$e ::= x \mid f(e_1, \ldots, e_n)$$

  where f maps into domain of values
- Boolean expression

$$b ::= x \mid f(e_1, \ldots, e_n)$$

  where f maps into Boolean domain

Programs

- (program) expression

$$e \ ::= \ x \mid f(e_1, \ldots, e_n)$$

  where f maps into domain of values

- Boolean expression

$$b \ ::= \ x \mid f(e_1, \ldots, e_n)$$

  where f maps into Boolean domain

- command

$$C \ ::= \ \textbf{skip} \mid x := e \mid C_1 \ ; \ C_2 \mid \textbf{if } b \textbf{ then } C_1 \textbf{ else } C_2 \mid \textbf{while } b \textbf{ do } C$$

Semantics of Expression e

- state s = function from program variables to value,

$$s : \mathbf{Var} \rightarrow \mathbf{Val}$$

Semantics of Expression e

- state s = function from program variables to value,

$$s : \mathbf{Var} \to \mathbf{Val}$$

- program expression e in state s evaluates to value

$$[\![e]\!](s) \in \mathbf{Val}$$

Semantics of Expression e

- state s = function from program variables to value,

$$s : \textbf{Var} \rightarrow \textbf{Val}$$

- program expression e in state s evaluates to value

$$[\![e]\!](s) \in \textbf{Val}$$

- semantics of program expressions e
  = function from set of states to set of values

$$[\![e]\!] : \textbf{States} \rightarrow \textbf{Val}$$

Semantics of Expression e

- state s = function from program variables to value,

$$s : \mathbf{Var} \to \mathbf{Val}$$

- program expression e in state s evaluates to value

$$[\![e]\!](s) \in \mathbf{Val}$$

- semantics of program expressions e
  = function from set of states to set of values

$$[\![e]\!] : \mathbf{States} \to \mathbf{Val}$$

- interpretation of function symbol f in expression $f(e_1, \ldots, e_n)$
  depends on logical first-order model
  ("+" interpreted over model of unbounded integers or
  in model for modulo arithmetic?)

Semantics of Boolean Expression b

- state s = function from program variables to values,

$$s : \textbf{Var} \rightarrow \textbf{Val}$$

Semantics of Boolean Expression b

- state s = function from program variables to values,

$$s : \mathbf{Var} \to \mathbf{Val}$$

- Boolean expression b in state s evaluates to Boolean truth value

$$[\![b]\!](s) \in \{\mathbf{T}, \mathbf{F}\}$$

Semantics of Boolean Expression b

- state s = function from program variables to values,

$$s : \mathbf{Var} \to \mathbf{Val}$$

- Boolean expression b in state s evaluates to Boolean truth value

$$[\![b]\!](s) \in \{\mathbf{T}, \mathbf{F}\}$$

- semantics of Boolean expression b
  = function from set of states to set of Boolean truth values

$$[\![b]\!] : \mathbf{States} \to \{\mathbf{T}, \mathbf{F}\}$$

Semantics of Boolean Expression b

- state s = function from program variables to values,

$$s : \mathbf{Var} \rightarrow \mathbf{Val}$$

- Boolean expression b in state s evaluates to Boolean truth value

$$[\![b]\!](s) \in \{\mathbf{T}, \mathbf{F}\}$$

- semantics of Boolean expression b
  = function from set of states to set of Boolean truth values

$$[\![b]\!] : \mathbf{States} \rightarrow \{\mathbf{T}, \mathbf{F}\}$$

- evaluation of Boolean expression b depends on logical first-order model
  ("$x \leq x + 1$" true in model of unbounded integers but false in model for modulo arithmetic)

Semantics of Commands C (1)

- semantics of command C
  $=$ functions from set of states to set of states

$$[\![C]\!] : \textbf{States} \rightarrow \textbf{States}, \quad s \mapsto s'$$

Semantics of Commands C (1)

- semantics of command C
  = functions from set of states to set of states

$$[\![C]\!] : \textbf{States} \rightarrow \textbf{States}, \quad s \mapsto s'$$

- execution of command C starting in state s ends in state s'

$$(C, s) \rightsquigarrow s'$$

Semantics of Commands C (1)

- semantics of command C
  = functions from set of states to set of states

$$[\![C]\!] : \textbf{States} \to \textbf{States}, \quad s \mapsto s'$$

- execution of command C starting in state s ends in state s'

$$(C, s) \leadsto s'$$

- execution of update statement
  = update of function $s : \textbf{Var} \to \textbf{Val}$

$$(x{:=}e, s) \leadsto s' \quad \text{where} \quad \begin{array}{l} s'(x) = [\![e]\!](s) \ \ \text{and} \\ s'(y) = s(y) \ \ \text{for} \ \ x \not\equiv y \end{array}$$

Semantics of Commands C (1)

- semantics of command C
  = functions from set of states to set of states

$$[\![C]\!] : \textbf{States} \to \textbf{States}, \quad s \mapsto s'$$

- execution of command C starting in state s ends in state s'

$$(C, s) \rightsquigarrow s'$$

- execution of update statement
  = update of function $s : \textbf{Var} \to \textbf{Val}$

$$(x{:=}e, s) \rightsquigarrow s' \quad \text{where} \quad \begin{array}{l} s'(x) = [\![e]\!](s) \quad \text{and} \\ s'(y) = s(y) \quad \text{for} \quad x \not\equiv y \end{array}$$

- execution of update depends on logical first-order model

Semantics of Commands C (2)

- execution of sequence of commands $C \equiv C_1 \, ; C_2$ $=$ execution of first command C_1 followed by execution of second command C_2

$$(C, s) \rightsquigarrow s'' \text{ if } (C_1, s) \rightsquigarrow s' \text{ and } (C_2, s') \rightsquigarrow s''$$

Semantics of Commands C (2)

- execution of sequence of commands $C \;\equiv\; C_1 \,;\, C_2$
  $=$ execution of first command C_1 followed by execution of second command C_2

$$(C, s) \rightsquigarrow s'' \;\text{ if }\; (C_1, s) \rightsquigarrow s' \;\text{ and }\; (C_2, s') \rightsquigarrow s''$$

- execution of command **skip** does not change state

$$(\mathbf{skip}, s) \rightsquigarrow s$$

("empty sequence of commands")

Semantics of Commands C (3)

- execution of conditional command $C \equiv$ **if** b **then** C_1 **else** C_2 $=$ execution of then-command C_1 if expression b evaluates to true

$$(C, s) \rightsquigarrow s' \quad \text{if} \quad [\![b]\!](s) = \mathbf{T} \quad \text{and} \quad (C_1, s) \rightsquigarrow s'$$

Semantics of Commands C (3)

- execution of conditional command $C \equiv$ **if** b **then** C_1 **else** C_2 $=$ execution of then-command C_1 if expression b evaluates to true

$$(C, s) \rightsquigarrow s' \quad \text{if} \quad [\![b]\!](s) = \mathbf{T} \quad \text{and} \quad (C_1, s) \rightsquigarrow s'$$

- execution of conditional command $C \equiv$ **if** b **then** C_1 **else** C_2 $=$ execution of then-command C_2 if expression b evaluates to false

$$(C, s) \rightsquigarrow s' \quad \text{if} \quad [\![b]\!](s) = \mathbf{F} \quad \text{and} \quad (C_2, s) \rightsquigarrow s'$$

Semantics of Commands C (3)

- execution of conditional command $C \equiv \textbf{if } b \textbf{ then } C_1 \textbf{ else } C_2$ $=$ execution of then-command C_1 if expression b evaluates to true

$$(C, s) \rightsquigarrow s' \quad \text{if} \quad [\![b]\!](s) = \mathbf{T} \quad \text{and} \quad (C_1, s) \rightsquigarrow s'$$

- execution of conditional command $C \equiv \textbf{if } b \textbf{ then } C_1 \textbf{ else } C_2$ $=$ execution of then-command C_2 if expression b evaluates to false

$$(C, s) \rightsquigarrow s' \quad \text{if} \quad [\![b]\!](s) = \mathbf{F} \quad \text{and} \quad (C_2, s) \rightsquigarrow s'$$

- execution of conditional depends on logical first-order model

Semantics of Commands C (4)

- execution of while command $C \equiv \textbf{while}\ b\ \textbf{do}\ C_0$
  $=$ execution of body C_0 followed by execution of while command C if expression b evaluates to true

  $(C, s) \rightsquigarrow s''$ if $[\![b]\!](s) = \mathbf{T}$ and $(C_0, s) \rightsquigarrow s'$ and $(C, s') \rightsquigarrow s''$

Semantics of Commands C (4)

- execution of while command $C \equiv \textbf{while}\ b\ \textbf{do}\ C_0$
  $=$ execution of body C_0 followed by execution of while command C if expression b evaluates to true

  $$(C, s) \rightsquigarrow s'' \text{ if } [\![b]\!](s) = \mathbf{T} \text{ and } (C_0, s) \rightsquigarrow s' \text{ and } (C, s') \rightsquigarrow s''$$

- execution of while command $C \equiv \textbf{while}\ b\ \textbf{do}\ C_0$
  $=$ execution of **skip** if expression b evaluates to false

  $$(C, s) \rightsquigarrow s \text{ if } [\![b]\!](s) = \mathbf{F}$$

Semantics of Commands C (4)

- execution of while command $C \equiv$ **while** b **do** C_0
  $=$ execution of body C_0 followed by execution of while command C if expression b evaluates to true

  $$(C, s) \rightsquigarrow s'' \ \text{ if } \ [\![b]\!](s) = \mathbf{T} \ \text{ and } \ (C_0, s) \rightsquigarrow s' \ \text{ and } \ (C, s') \rightsquigarrow s''$$

- execution of while command $C \equiv$ **while** b **do** C_0
  $=$ execution of **skip** if expression b evaluates to false

  $$(C, s) \rightsquigarrow s \ \text{ if } \ [\![b]\!](s) = \mathbf{F}$$

- execution of while loop depends on logical first-order model

Hoare Triple $\{\phi\}\ C\ \{\psi\}$

- $\{\phi\}\ C\ \{\psi\}$ valid in given logical first-order model if

Hoare Triple $\{\phi\}\ C\ \{\psi\}$

- $\{\phi\}\ C\ \{\psi\}$ valid in given logical first-order model if
  for all states s
  if $[\![\phi]\!](s) = \mathbf{T}$ and

Hoare Triple $\{\phi\}\ C\ \{\psi\}$

- $\{\phi\}\ C\ \{\psi\}$ valid in given logical first-order model if
  for all states s
  if $[\![\phi]\!](s) = \mathbf{T}$ and
  if $(C, s) \rightsquigarrow s'$ then

Hoare Triple $\{\phi\}\ C\ \{\psi\}$

- $\{\phi\}\ C\ \{\psi\}$ valid in given logical first-order model if
  for all states s
  if $[\![\phi]\!](s) = \mathbf{T}$ and
  if $(C, s) \leadsto s'$ then
  $[\![\psi]\!](s') = \mathbf{T}$
- $\{\phi\}\ C\ \{\psi\}$ valid if valid in every logical first-order model
- $\Gamma \models \{\phi\}\ C\ \{\psi\}$ if $\{\phi\}\ C\ \{\psi\}$ valid in every logical first-order model of set of assertions Γ

Variables in Hoare Triple $\{\phi\}$ C $\{\psi\}$

- program variables: occur in commands in program C

Variables in Hoare Triple $\{\phi\}\ C\ \{\psi\}$

- program variables: occur in commands in program C
  may occur (*free*) in ϕ and ψ
- auxiliary variables: occur (*free*) in ϕ and/or ψ
  but do not occur in commands in program C

Variables in Hoare Triple $\{\phi\}$ C $\{\psi\}$

- program variables: occur in commands in program C may occur (*free*) in ϕ and ψ
- auxiliary variables: occur (*free*) in ϕ and/or ψ but do not occur in commands in program C
- needed, e.g., for specification of *in-place sort* program

$$\textbf{if } x \leq y \textbf{ then skip else } z := y \;;\; y := x \;;\; x := z$$

Variables in Hoare Triple $\{\phi\}\ C\ \{\psi\}$

- program variables: occur in commands in program C
  may occur (*free*) in ϕ and ψ
- auxiliary variables: occur (*free*) in ϕ and/or ψ
  but do not occur in commands in program C
- needed, e.g., for specification of *in-place sort* program

$$\textbf{if}\ x \le y\ \textbf{then skip else}\ z := y\ ;\ y := x\ ;\ x := z$$

- take precondition $\phi\ \equiv\ x = x_0 \wedge y = y_0 \wedge x_0 > y_0$
  and postcondition $\psi\ \equiv\ x = y_0 \wedge y = x_0$